

**Xml第一次作业**

**关于GB2312，big5,UTF\_8/16,unicode编码集**

**系 （院）：**  计算机科学学院

**专业班级：** 软工1501

**姓 名：** 周彦

**学 号：** 201500562

**指导教师：** 向华

**目 录**

1. GB2312

1.1概述....................................................................................................................................3

1.2字节结构............................................................................................................................3

1.3分区表示............................................................................................................................3

1. Big5

2.1概述....................................................................................................................................3

2.2产生背景............................................................................................................................3

2.3字节结构............................................................................................................................3

2.4私人造字区........................................................................................................................3

2.5发展....................................................................................................................................4

1. UTF-8

3.1概述....................................................................................................................................4

3.2编码字节含义....................................................................................................................4

3.3设计原则............................................................................................................................4

3.4编码方式.............................................................................................................................4

1. UTF-16

4.1概述.....................................................................................................................................5

4.2编码方式.............................................................................................................................5

1. Unicode

5.1概述.....................................................................................................................................5

5.2编码方式.............................................................................................................................5

5.3 XML和Unicode...................................................................................................................5

5.4动态组字............................................................................................................................6

六、编码集的相互转化

6.1 UTF-8转UTF-16...................................................................................................................6

6.2 UTF-16转UT-8....................................................................................................................7

注明：以下内容均来自wikipedia

一、GB2312

**概述：**GB 2312 或 GB 2312–80 是中华人民共和国国家标准简体中文字符集合，全称《信息交换用汉字编码字符集·基本集》，又称GB0，由中国国家标准总局发布，1981年5月1日实施。GB 2312编码通行于中国大陆；新加坡等地也采用此编码。中国大陆几乎所有的中文系统和国际化的软件都支持GB 2312。

**字节结构：**在使用GB 2312的程序通常采用EUC储存方法，以便兼容于ASCII。这种格式称为EUC-CN。浏览器编码表上的“GB2312”就是指这种表示法。每个汉字及符号以两个字节来表示。第一个字节称为“高位字节”，第二个字节称为“低位字节”。 “高位字节”使用了0xA1–0xF7（把01–87区的区号加上0xA0），“低位字节”使用了0xA1–0xFE（把01–94加上0xA0）。 由于一级汉字从16区起始，汉字区的“高位字节”的范围是0xB0–0xF7，“低位字节”的范围是0xA1–0xFE，占用的码位是72\*94=6768。其中有5个空位是D7FA–D7FE。

**分区表示：**GB 2312中对所收汉字进行了“分区”处理，每区含有94个汉字／符号。这种表示方式也称为区位码。01-09区为特殊符号。16-55区为一级汉字，按拼音排序。56-87区为二级汉字，按部首笔画排序。

二、big5

**概述：**Big5，又称为大五码或五大码，是使用繁体中文（正体中文）社区中最常用的电脑汉字字符集标准，共收录13,060个汉字。2003年，Big5被收录到CNS11643中文标准交换码的附录当中，获取了较正式的地位。这个最新版本被称为Big5-2003。

**产生背景：**Big5码的产生，是因为当时个人电脑没有共通的内码，导致厂商推出的中文应用软件无法推广，并且与IBM 5550、王安码等内码，彼此不能兼容；另一方面，台湾当时尚未推出中文编码标准。在这样的时空背景下，为了使台湾早日进入信息时代，所采行的一个项目；同时，这个项目对于以台湾为核心的亚洲繁体汉字圈也产生了久远的影响。Big5产生前，研发中文电脑的朱邦复认为内码字集应该广纳所有的正异体字，以顾及如户政等应用上的需要，故在当时的内码会议中，建议希望采用他的五万多字的字库。工程师认为虽其技术可行，但是三个字节（超过两个字节）长度的内码却会造成英文显示屏画面映射成中文画面会发生文字无法对齐的问题，因为当时盛行之倚天中文系统画面系以两个字节文字宽度映射成一个中文字图样，英文软件中只要以两个英文字宽度去显示一个中文字，画面就不会乱掉，造成中文系统业者偏爱二个字节长度的内码；此外以仓颉输入码压缩成的内码不具排序等功能，因此未被采用。1983年有人诬指朱邦复为共产党，其研究成果更不可能获采用。在Big5码诞生后，大部分台湾的电脑软件都使用了Big5码，加上后来倚天中文系统的高度普及，使后来的微软Windows 3.x等亦予以采用。虽然后来台湾还有各种想要取代Big5码，像是倚天中文系统所推行的倚天码、台北市电脑公会所推动的公会码等，但是由于Big5字码已沿用多年，因此在习惯不易改变的情况下，始终无法成为主流字码。而台湾后来发展的国家标准CNS 11643中文标准交换码由于非一般的内码系统，是以交换使用为目的，受先天所限，必须使用至少三个字节来表示一个汉字，所以普及率远远不及Big5码。

**字节结构：**Big5码是一套双字节字符集，使用了双八码存储方法，以两个字节来安放一个字。第一个字节称为“高位字节”，第二个字节称为“低位字节”。 “高位字节”使用了0x81-0xFE，“低位字节”使用了0x40-0x7E，及0xA1-0xFE。

**私人造字区：**在倚天中文系统，以及后来的Windows 3.1、95及98中，定义四个私人造字区范围：0xFA40-0xFEFE、0x8E40-0xA0FE、0x8140-0x8DFE、0xC6A1-0xC8FE。私人造字区的原意，是供用户加入本来在编码表中缺少的字符，但当每个用户都在不同的地方加上不同的字符后，当交换数据时，对方便难以知道某一个编码究竟想表达什么字。

**发展：**由于各厂商及政府推出的Big5延伸，彼此互不兼容，造成乱码问题。鉴于Unicode能正确地处理七万多个汉字，近年的操作系统和应用程序（如苹果电脑Mac OS X和以Cocoa API撰写之程序、Microsoft Windows 2000及之后版本、Microsoft Office 2000及之后版本、Mozilla浏览器、、Java语言等等），已改用Unicode编码。可惜现时仍有一些旧的软件（如部分Telnet或BBS软件），未能支持Unicode编码，故相信Big5缺字的问题仍会困扰用户一段时间，直至所有程序都能改用Unicode为止。

三、UTF-8

**概述：UTF-8**（**8-bit Unicode Transformation Format**）是一种针对Unicode的可变长度字符编码，也是一种前缀码。它可以用来表示Unicode标准中的任何字符，且其编码中的第一个字节仍与ASCII兼容，这使得原来处理ASCII字符的软件无须或只须做少部分修改，即可继续使用。因此，它逐渐成为电子邮件、网页及其他存储或发送文字的应用中，优先采用的编码。

**UTF-8编码字节含义：**

* 对于UTF-8编码中的任意字节B，如果B的第一位为0，则B独立的表示一个字符(ASCII码)；
* 如果B的第一位为1，第二位为0，则B为一个多字节字符中的一个字节(非ASCII字符)；
* 如果B的前两位为1，第三位为0，则B为两个字节表示的字符中的第一个字节；
* 如果B的前三位为1，第四位为0，则B为三个字节表示的字符中的第一个字节；
* 如果B的前四位为1，第五位为0，则B为四个字节表示的字符中的第一个字节；

因此，对UTF-8编码中的任意字节，根据第一位，可判断是否为ASCII字符；根据前二位，可判断该字节是否为一个字符编码的第一个字节；根据前四位（如果前两位均为1），可确定该字节为字符编码的第一个字节，并且可判断对应的字符由几个字节表示；根据前五位（如果前四位为1），可判断编码是否有错误或数据传输过程中是否有错误

**设计UTF-8的理由：**

UTF-8的设计有以下的多字符组序列的特质：

* 单字节字符的最高有效比特永远为0。
* 多字节序列中的首个字符组的几个最高有效比特决定了序列的长度。最高有效位为110的是2字节序列，而1110的是三字节序列，如此类推。
* 多字节序列中其余的字节中的首两个最高有效比特为10。

UTF-8的这些特质，保证了一个字符的字节序列不会包含在另一个字符的[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)序列中。这确保了以字节为基础的部分字符串比对（sub-string match）方法可以适用于在文字中搜索字或词。有些比较旧的可变长度8位编码没有这个特质，故字符串比对的算法变得相当复杂。虽然这增加了UTF-8编码的字符串但是利多于弊。另外，数据压缩并非Unicode的目的，所以不可混为一谈。即使在发送过程中有部分字节因错误或干扰而完全丢失，还是有可能在下一个字符的起点重新同步，令受损范围受到限制。另一方面，由于其字节序列设计，如果一个疑似为字符串的序列被验证为UTF-8编码，那么我们可以有把握地说它是UTF-8字符串。一段两字节随机序列碰巧为合法的UTF-8而非ASCII的概率为32分1。对于三字节序列的概率为256分1，对更长的序列的概率就更低了。

**UTF-8编码方式：**在ASCII码的范围，用一个字节表示，超出ASCII码的范围就用字节表示，这就形成了我们上面看到的UTF-8的表示方法，这様的好处是当UNICODE文件中只有ASCII码时，存储的文件都为一个字节，所以就是普通的ASCII文件无异，读取的时候也是如此，所以能与以前的ASCII文件兼容。大于ASCII码的，就会由上面的第一字节的前几位表示该unicode字符的长度，比如110xxxxx前三位的二进制表示告诉我们这是个2BYTE的UNICODE字符；1110xxxx是个三位的UNICODE字符，依此类推；xxx的位置由字符编码数的二进制表示的位填入。越靠右的x具有越少的特殊意义。只用最短的那个足够表达一个字符编码数的多字节串。注意在多字节串中，第一个字节的开头"1"的数目就是整个串中字节的数目。

四、UTF-16：

**概述：**UTF-16是Unicode字符编码五层次模型的第三层：字符编码表（Character Encoding Form，也称为"storage format"）的一种实现方式。即把Unicode字符集的抽象码位映射为16位长的整数（即码元）的序列，用于数据存储或传递。Unicode字符的码位，需要1个或者2个16位长的码元来表示，因此这是一个变长表示。

**UTF-16的编码方式：**UTF-16的大尾序和小尾序存储形式都在用。一般来说，以Macintosh制作或存储的文字使用大尾序格式，以Microsoft或Linux制作或存储的文字使用小尾序格式。为了弄清楚UTF-16文件的大小尾序，在UTF-16文件的开首，都会放置一个U+FEFF字符作为Byte Order Mark（UTF-16LE以FF FE代表，UTF-16BE以FE FF代表），以显示这个文本文件是以UTF-16编码，其中U+FEFF字符在UNICODE中代表的意义是ZERO WIDTH NO-BREAK SPACE，顾名思义，它是个没有宽度也没有断字的空白。

五、unicode：

**概述：**Unicode（中文：万国码、国际码、统一码、单一码）是计算机科学领域里的一项业界标准。它对世界上大部分的文字系统进行了整理、编码，使得电脑可以用更为简单的方式来呈现和处理文字。Unicode涵盖的数据除了视觉上的字形、编码方法、标准的字符编码外，还包含了字符特性，如大小写字母。Unicode备受认可，并广泛地应用于电脑软件的国际化与本地化过程。有很多新科技，如可扩展置标语言(Extensible Markup Language，简称：XML)、Java编程语言以及现代的操作系统，都采用Unicode编码

**编码方式：**统一码的编码方式与ISO 10646的通用字符集概念相对应。目前实际应用的统一码版本对应于UCS-2，使用16位的编码空间。也就是每个字符占用2个字节。这样理论上一共最多可以表示216（即65536）个字符。基本满足各种语言的使用。实际上当前版本的统一码并未完全使用这16位编码，而是保留了大量空间以作为特殊使用或将来扩展。上述16位统一码字符构成基本多文种平面。最新（但未实际广泛使用）的统一码版本定义了16个辅助平面，两者合起来至少需要占据21位的编码空间，比3字节略少。但事实上辅助平面字符仍然占用4字节编码空间，与UCS-4保持一致。未来版本会扩充到ISO 10646-1实现级别3，即涵盖UCS-4的所有字符。UCS-4是一个更大的尚未填充完全的31位字符集，加上恒为0的首位，共需占据32位，即4字节。理论上最多能表示231个字符，完全可以涵盖一切语言所用的符号。基本多文种平面的字符的编码为*U+hhhh*，其中每个*h*代表一个十六进制数字，与UCS-2编码完全相同。而其对应的4字节UCS-4编码后两个字节一致，前两个字节则所有位均为0。关于统一码和ISO 10646及UCS的详细关系，见通用字符集。

**Xml和unicode：**XML及其子集XHTML采用UTF-8作为标准字集，理论上我们可以在各种支持XML标准的浏览器上显示任何地区文字的网页，只要电脑本身安装有合适的字体即可。可以利用&#nnn;的格式显示特定的字符。nnn代表该字符的十进制Unicode代码。如果采用十六进制代码，在编码之前加上x字符即可。但部分旧版本的浏览器可能无法识别十六进制代码。过去电脑编码的8位标准，使每个国家都只按国家使用的字符而编定各自的编码系统；而对于部分字符系统比较复杂的语言，如越南语，又或者东亚国家的大型字符集，都不能在8位的环境下正常显示。只是最近才有在文本中对十六进制的支持，那么旧版本的浏览器显示那些字符或许可能有问题-大概首先会遇到的一个问题只是在对于大于8位Unicode字符的显示。解决这个问题的普遍做法仍然是将其中的十六进制码转换成一个十进制码也有一些字符集标准将一些常用的标志存放在字符内码外面，那么你可能使用象—这样的文本标志来表示一个长划（—）的情况，即使它的字符内码已经被使用，这些标准也不包含那个字符。然而部分由于Unicode版本发展原因，很多浏览器只能显示UCS-2完整字符集，也即现在使用的Unicode版本中的一个小子集。一些多语言支持的网页浏览器，比如微软Windows系统的Internet Explorer5.5及以上版本，以及跨平台的浏览器Mozilla／Netscape 6，可以在安装时根据需要动态地使用相应的字符集，预先安装了合适的语言包，就可以同时显示页面上的各种Unicode字符。Internet Explorer 5.5还提出用户可以在需要新字体时，即装即用。另外的浏览器如Netscape Navigator 4.77，则只能显示跟页面编码相应字符集中的文字。当你使用后一种浏览器时，你不大可能预先安装所有的字体，即使有了字体，浏览器也不一定能将这些字体完全应用起来。可能遇到的情况是，这种浏览器只能够显示部分文字，因为它们是按照标准进行编码，尽管理论上在兼容的系统中，只要有了相应的Code2000字体，就可以正确显示。一种变通的办法，是将某些少见的字符，通过“名称实体引用”的方式来使用。

**动态组字：**统一码这种为数万汉字逐一编码的方式很浪费资源，且要把汉字增加到标准中也并不容易，因此去研究以汉字部件产生汉字的方法（如[动态组字](https://zh.wikipedia.org/wiki/%E5%8B%95%E6%85%8B%E7%B5%84%E5%AD%97)），期望取代为汉字逐一编码的方法。Unicode委员会在关于中文和日语的常用问题列表[[28]](https://zh.wikipedia.org/wiki/Unicode#cite_note-han_cjk-29)里回答了此问题。主要问题是汉字中各个组件的相对大小不是固定的。比如“员”字，由“口”和“贝”组成，而“呗”也是由“口”和“贝”组成，但其相对位置和大小并不一致。还有一些其他原因，比如字符比较和排序时需要先对编码流进行分析后才能得到各个字符，增加处理程序复杂性等。

六、编码集的相互转化：

 **6.1    UTF-16转UTF-8：**

UTF16和UTF8之间的相互转换可以通过上图的转换表来实现，判断Unicode码所在的区间就可以得到这个字符是由几个字节所组成，之后通过移位来实现,分为新的多个字节来存储。

Step1:获取该字符对应的Unicode码。

Step2:判断该Unicode码所在的范围,根据不同的范围,来决定存储它的字节长度。

如果介于U+00000000 – U+0000007F之间,代表该字符采取一个字节存储,那么直接通过这个新字节的unicode码,即可转换为UTF-8码(这是这里的一种简称,不同的编程语言有不同实现,例如可以用两个字节来存储一个字符的信息,解码时进行判断,如果发现是UTF-8的多字节实现,那么将多字节合并后再转为一个字符输出).转换完毕

如果介于U+00000080 – U+000007FF之间,代表该字符采取两个字节存储,那么将该Unicode码转为二进制,取出高5位(这里不分大端序和小端序，只以实际的码为准，具体实现可以采取移位实现)，并加上头部110，组成第一个字节；再取出低6位(按顺序取),加上头部10，组成第二个字节。然后分别通过两个新的字节的unicode码,可以转换为相应的UTF-8码.转换完毕。

如果介于U+00000800 – U+0000FFFF之间,代表该字符采取三个字节存储,那么将该Unicode码转为二进制,取出高4位，并加上头部1110，组成第一个字节；再取出低6位(按顺序取),加上头部10，组成第二个字节；再取出低6位(按顺序取),加上头部10，组成第三个字节。然后分别通过三个新的字节的unicode码,可以转换为相应的UTF-8码.转换完毕。

如果介于U+00010000 – U+001FFFFF之间,代表该字符采取四个字节存储(实际上,四个字节或以上存储的字符是很少的),那么将该Unicode码转为二进制,取出高3位，并加上头部11110，组成第一个字节；再取出低6位(按顺序取),加上头部10，组成第二个字节；再取出低6位(按顺序取),加上头部10，组成第三个字节；再取出低6位(按顺序取),加上头部10，组成第四个字节。然后分别通过四个新的字节的unicode码,可以转换为相应的UTF-8码.转换完毕。

如果介于U+00200000 – U+03FFFFFF,代表该字符采取五个字节存储,那么将该Unicode码转为二进制,取出高2位，并加上头部111110，组成第一个字节；再取出低6位(按顺序取),加上头部10，组成第二个字节；再取出低6位(按顺序取),加上头部10，组成第三个字节；再取出低6位(按顺序取),加上头部10，组成第四个字节；再取出低6位(按顺序取),加上头部10，组成第五个字节。然后分别通过五个新的字节的unicode码,可以转换为相应的UTF-8码.转换完毕。

如果介于U+04000000 – U+7FFFFFFF,代表该字符采取六个字节存储,那么将该Unicode码转为二进制,取出高1位，并加上头部1111110，组成第一个字节；再取出低6位(按顺序取),加上头部10，组成第二个字节；再取出低6位(按顺序取),加上头部10，组成第三个字节；再取出低6位(按顺序取),加上头部10，组成第四个字节；再取出低6位(按顺序取),加上头部10，组成第五个字节；再取出低6位(按顺序取),加上头部10，组成第六个字节。然后分别通过六个新的字节的unicode码,可以转换为相应的UTF-8码.转换完毕。

#### 6.2：UTF-8转UTF-16

#### 这是UTF8转换到UTF16，同样需要对照转换表来实现。需要判断字符的Unicode码,然后判断是属于用几个字节存储的,然后分别用对应的字节数合并起来,形成新的字符输出。

#### Step1:获取该字符对应的Unicode码。

#### Step2:用该码的二进制和相应的关键字节相与,根据上图,判断处于那一段区间,来判断是使用几个字节存储字符的,然后分别合并对应的字节数,组成新的字符输出。

#### 用该Unicode码的二进制右移7位后与(11111111)相与,如果得到了0,代表该字符只用了一个字节存储,所以直接输出该字符.转换完毕。

用该Unicode码的二进制右移5位后与(11111111)相与，如果得到了110(6),代表该字符占用了二个字节,所以分别获取该字符和下一个字符,然后分别取出本字节的低5位后左移6位和取出下一个字节的低6位(保持不变)，将2个字节相或，得到一个新的字节.这个字节就是最终字符的unicode码,然后转为对应的字符输出.转换完毕。

用该Unicode码的二进制右移4位后与(11111111)相与，如果得到了1110(14),代表该字符占用了三个字节,所以分别获取该字符和下一个字符和下下个字符,然后分别取出本字节的低4位后左移12位和取出下一个字节的低6位后左移6位和取出下下一个字节的低6位(保持不变)，将3个字节相或，得到一个新的字节.这个字节就是最终字符的unicode码,然后转为对应的字符输出. 转换完毕。

用该Unicode码的二进制右移3位后与(11111111)相与，如果得到了11110(30),代表该字符占用了四个字节,所以分别获取该字符和下一个字符和下下个字符和下下下个字符,然后分别取出本字节的低3位后左移18位取出下一个字节的低6位后左移12位和和取出下下一个字节的低6位后左移6位和取出下下下一个字节的低6位(保持不变)，将4个字节相或，得到一个新的字节.这个字节就是最终字符的unicode码,然后转为对应的字符输出. 转换完毕。

用该Unicode码的二进制右移2位后与(11111111)相与，如果得到了111110(62),代表该字符占用了五个字节,所以分别获取该字符和下一个字符和下下个字符和下下下个字符和下下下下个字符,然后分别取出本字节的低2位后左移24位和取出下一个字节的低6位后左移18位和取出下下一个字节的低6位后左移12位和和取出下下下一个字节的低6位后左移6位和取出下下下下一个字节的低6位(保持不变)，将5个字节相或，得到一个新的字节.这个字节就是最终字符的unicode码,然后转为对应的字符输出. 转换完毕。

用该Unicode码的二进制右移1位后与(11111111)相与，如果得到了1111110(126),代表该字符占用了六个字节,所以分别获取该字符和下一个字符和下下个字符和下下下个字符和下下下下个字符和下下下下下个字符,然后分别取出本字节的低1位后左移30位和取出下一个字节的低6位后左移24位和取出下下一个字节的低6位后左移18位和取出下下下一个字节的低6位后左移12位和和取出下下下下一个字节的低6位后左移6位和取出下下下下下一个字节的低6位(保持不变)，将6个字节相或，得到一个新的字节.这个字节就是最终字符的unicode码,然后转为对应的字符输出. 转换完毕。